



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/872,586	06/01/2001	Kevin Alexander Stoodley	CA920000035US1	2632
25259	7590	04/05/2004	EXAMINER	
IBM CORPORATION			TANG, KUO LIANG J	
3039 COWNALLIS RD.				
DEPT. T81 / B503, PO BOX 12195				
REASEARCH TRIANGLE PARK, NC 27709			ART UNIT	PAPER NUMBER
			2122	
DATE MAILED: 04/05/2004				

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/872,586	Applicant(s) STOODLEY, KEVIN ALEXANDER
	Examiner Kuo-Liang J Tang	Art Unit 2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
 - If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
 - If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
 - Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 01 June 2001.
- 2a) This action is FINAL. 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 1-30 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 1-30 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 a) All b) Some * c) None of:
 1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) Notice of References Cited (PTO-892)
- 2) Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
 Paper No(s)/Mail Date _____
- 4) Interview Summary (PTO-413)
 Paper No(s)/Mail Date _____
- 5) Notice of Informal Patent Application (PTO-152)
- 6) Other: _____

1. Claims 1-30 are pending and have been examined. The priority date for this application is 01/23/2001

Specification

2. The abstract of the disclosure is objected to because the abstract of the disclosure exceeds 150 words in length. Correction is required. See MPEP § 608.01(b).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

3. Claims 1-5, 11-18 and 21-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Holzle et al., "Optimizing Dynamically-Typed Object-Oriented Languages With Polymorphic Inline Caches", In Proceedings of the 5th European Conference on Object-Oriented Programming – ECOOP '91, volume 512 of Lecture Notes in Computer Science, Springer-Verlag, pp. 21-38, 1991 (hereinafter Holzle) in view of Bacon US Patent No. 6,247,025.

As Per Claim 1, Holzle teaches that Polymorphic inline caches (PICs) provide a new way to reduce the overhead of polymorphic message sends by extending inline caches to include more than one cached lookup result per call site. (E.g. see Abstract and associated text). In that Holzle discloses the method that covering the steps of:

"a) creating a template of a polymorphic inline cache for a polymorphic call site, the template having a plurality of slots," (E.g. see Figure 3., PIC stub and associated text. I.E. page 4

last paragraph, which states “... Eventually, the stub will contain all cases seen in practice, and ...”, the Examiner interprets the “cases” are “plurality of slots”, each case is one slot.);

“b) executing the polymorphic inline cache with an object of type k;” (E.g. see Figure 3., PIC stub, ‘type = rectangle” and associated text. the Examiner interprets the “rectangle” is “type k”);

“c) invoking a polymorphic inline cache initialisation routine,” (E.g. see Figure 3. PIC stub, “if type = rectangle jump to method” and associated text.”);

“d) finding an available k slot of the polymorphic inline cache;” (E.g. see Figure 3. PIC stub, “if type = rectangle jump to method” and associated text.”);

“f) searching for a k method to call for the object of type k;” (E.g. see Figure 3. PIC stub, “jump to method” and associated text. For each type has its own method.”);

“g) filling the k slot with a call instruction to the k method;” (E.g. see Figure 3. rectangle display method and associated text.”); and

“i) calling the k method of the object of type k.” (E.g. see Figure 3. PIC stub, “if type = rectangle jump to method” and associated text.”).

Holzle does not explicitly disclose in a multi-thread system that e) locking the k slot of the polymorphic inline cache and h) unlocking the k slot to complete the k slot. Holzle discloses SmallTalk (Object-Oriented) (E.g. see page 8, line 26) that suggests avoiding issues from contention between simultaneously executing processes. In that instance of object created operates in its own namespace. That is analogous to a multi-threading environment. Bacon

teaches a lock/unlock mechanism to control concurrent access to objects in a multi-threaded computer processing system (E.g. see Abstract and associated text). Bacon discloses "The operating system 108 of the present invention provides multi-threading capabilities wherein multiple concurrent threads of control are dispatched within a single shared address space." (E.g. see col. 4:53-63, locking mechanism). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bacon into the system of Holzle, to lock the k slot of the polymorphic inline cache when the method is invoked and unlock the k slot of the polymorphic inline cache when the method is finished in a multi-threading object-oriented environment. The modification would have been obvious because one of ordinary skill in the art would have been motivated to solve a synchronization problem, and to ensure that two threads are not trying to make changes to the same memory location.

As Per claim 2, the rejection of claim 1 is incorporated and further the limitation of "a first thread may initialize and/or access the k slot at the same time a second thread is initializing and/or accessing a slot other than the k slot." is obvious in view of the combination of Holzle and Bacon as noted above in Claim 1 which is a PIC in a multi-threading environment.

As Per claim 3, the rejection of claim 2 is incorporated and further the limitation of "a second thread calling the k method while the first thread is initializing the k slot waits until after said unlocking step (h)." is obvious as noted above in Claim 1.

Art Unit: 2122

As Per claim 4, the rejection of claim 2 is incorporated and further the limitation of “a second thread calling the k method while the first thread is initializing the k slot searches for and calls the k method and leaves the polymorphic inline cache unchanged.” is obvious because the slot k is locked by first thread.

As Per claim 5, the rejection of claim 1 is incorporated and further Holzle discloses “a polymorphic inline cache is created for each polymorphic call site.”(E.g. see Figure 3., PIC stub and associated text. I.E. page 4 last paragraph, which states “... Eventually, the stub will contain all cases seen in practice, and ...”, the Examiner interprets the “cases” are “plurality of slots”, each case is one slot.).

As Per claim 11, the rejection of claim 1 is incorporated and further Holzle discloses “the step of (j) updating the polymorphic inline cache so that an object of type k+1 will initialise a corresponding k+1 slot.” (E.g. see Figure 3. PIC stub and associated text, i.e. type k for rectangle, and type k+1 for circle).

As Per claim 12, the rejection of claim 11 is incorporated and further Holzle discloses “said updating step (j) further comprises inspecting the polymorphic inline cache to find the next empty slot.” (E.g. see Figure 3. circle display method and associated text).

As Per claim 13, the rejection of claim 11 is incorporated and further Holzle discloses “said polymorphic inline cache initialisation routine is the same for every object of the polymorphic inline cache.” (E.g. see Figure 3. circle display method and associated text).

As Per claim 14, the rejection of claim 11 is incorporated and further Holzle discloses “said updating step (j) further comprises maintaining a state of the k slot or the k+1 slot.” (E.g. see Figure 3. PIC stub and associated text, i.e. type k for rectangle, and type k+1 for circle).

As Per claim 15, the rejection of claim 14 is incorporated and further Holzle discloses “said polymorphic inline cache initialisation routine is the same for every object of the polymorphic inline cache.” (E.g. see Figure 3. circle display method and associated text).

As Per claim 16, the rejection of claim 15 is incorporated and further Holzle discloses “said updating step (j) further comprises maintaining a state of the k slot or the k+1 slot.” (E.g. see Figure 3. PIC stub and associated text, i.e. type k for rectangle, and type k+1 for circle).

As Per claim 17, the rejection of claim 1 is incorporated and further Holzle discloses “said invoking step (c) further comprises calling a different initialization routine for every object of a different type.” (E.g. see Figure 3. PIC stub and associated text, i.e. type rectangle and circle).

As Per claim 18, the rejection of claim 11 is incorporated and further Holzle discloses “said updating step (j) further comprises modifying the initialization routine so that upon a cache miss of the k slot, the k+1 initialization routine is called.” (E.g. see Figure 3. PIC stub and associated text, i.e. type k+1 for circle).

As Per claim 21, Holzle discloses “calling a first method having a first object type from an executing object oriented program;” (E.g. see Figure 3. PIC stub and rectangle display method and associated text). and “executing the first slot of the polymorphic inline cache.” (E.g. see Figure 3. PIC stub and rectangle display method and associated text).

Holzle does not explicitly disclose locking a first slot of the polymorphic inline cache with a call to the first method of the first object type. Holzle discloses SmallTalk (Object-Oriented) (E.g. see page 8, line 26) that suggests avoiding issues from contention between simultaneously executing processes. In that instance of object created operates in its own namespace. That is analogous to a multi-threading environment. Bacon teaches a lock/unlock mechanism to control concurrent access to objects in a multi-threaded computer processing system (E.g. see Abstract and associated text). Bacon discloses “The operating system 108 of the present invention provides multi-threading capabilities wherein multiple concurrent threads of control are dispatched within a single shared address space.” (E.g. see col. 4:53-63, locking mechanism). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bacon into the system of Holzle, to locking a first slot of the polymorphic inline cache with a call to the first method of the first

object type in a multi-threading object-oriented environment. The modification would have been obvious because one of ordinary skill in the art would have been motivated to solve a synchronization problem, and to ensure that two threads are not trying to make changes to the same memory location.

As Per claim 22, the rejection of claim 21 is incorporated and further Holzle discloses “calling a second method having a second object type;” (E.g. see Figure 3. PIC stub and circle display method and associated text).

“locking a second slot of the polymorphic inline cache with a call to the second method of the second object type;” (as noted above in Claim 21) and

“executing the second slot of the polymorphic inline cache.” (E.g. see Figure 3. PIC stub and circle display method and associated text).

As Per claim 23, the rejection of claim 22 is incorporated and further the limitation “the first slot of the polymorphic inline cache is executed simultaneously with the second slot of the polymorphic inline cache.” is obvious in a multi-threading environment as noted above in Claim 21.

As Per claim 24, the rejection of claim 22 is incorporated and further Holzle discloses “the first method (E.g. see Figure 3. PIC stub and rectangle display method and associated text) of the first object type is called from a first thread which executes independently

from an executing second thread which called the second method (E.g. see Figure 3. PIC stub and circle display method and associated text) of the second object type.”

As Per claim 25, the rejection of claim 21 is incorporated and further Holzle discloses “calling a second method having a second object type,” (E.g. see Holzle Figure 3. PIC stub and circle display method and associated text);

“waiting until the first method of the first object type has executed;” (E.g. see Bacon col. 1:24-32, wait);

“determining whether the second object type and the first object type are the same object type;” (E.g. see Holzle Figure 3. PIC stub and associated text); and

“not locking a second slot of the polymorphic inline cache with a call to the method of the second type if the first and second object types are the same object type;” (E.g. see Bacon col. 1:24-32) and

“locking the second slot with a call to the second method if the first and second object types are not the same object type and executing the second slot.” (E.g. see Bacon col. 1:24-32).

As Per claim 26, Holzle discloses

“means for executing an object oriented program” (E.g. see Title and associated text);

“means for calling a first method from a first slot of a polymorphic inline cache” (E.g. see Figure 3. PIC stub and rectangle display method and associated text);

“means for calling a second method from a second slot of the polymorphic inline cache”

(E.g. see Figure 3. PIC stub and circle display method and associated text);

Art Unit: 2122

“means for determining if the first method and the second method have an identical object type;” (E.g. see Figure 3. PIC stub, type checking and associated text);

“means for calling the first method and the second method simultaneously if they do not have the identical object type;” (E.g. see Figure 3. PIC stub and rectangle and circle display methods and associated text); and

“means for preventing calling the second method from the second slot of the polymorphic inline cache until said means for calling the first method has completed if they have the identical object type.”

Holzle does not explicitly disclose means for preventing calling the second method from the second slot of the polymorphic inline cache until said means for calling the first method has completed if they have the identical object type. Holzle discloses SmallTalk (Object-Oriented) (E.g. see page 8, line 26) that suggests avoiding issues from contention between simultaneously executing processes. In that instance of object created operates in its own namespace. That is analogous to a multi-threading environment. Bacon teaches a lock/unlock mechanism to control concurrent access to objects in a multi-threaded computer processing system (E.g. see Abstract and associated text). Bacon discloses “The operating system 108 of the present invention provides multi-threading capabilities wherein multiple concurrent threads of control are dispatched within a single shared address space.” (E.g. see col. 4:53-63, locking mechanism). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bacon into the system of Holzle, to prevent calling the second method from the second slot of the polymorphic inline cache until said means for calling the first method has completed if they have the identical object type in a multi-threading object-

oriented environment. The modification would have been obvious because one of ordinary skill in the art would have been motivated to solve a synchronization problem, and to ensure that two threads are not trying to make changes to the same memory location.

As Per claim 27, the rejection of claim 26 is incorporated and further the limitation “the first slot of the polymorphic inline cache is executed simultaneously with the second slot of the polymorphic inline cache.” is obvious in a multi-threading environment as noted above in Claim 26.

As Per claim 28, the rejection of claim 27 is incorporated and further Holzle discloses “means for invoking a first polymorphic inline cache initialisation routine;” (E.g. see Figure 3., PIC stub and associated text.);

“means for locking the first slot of the polymorphic inline cache;” (as noted above in Claim 26, lock);

“means for filling the first slot with a call instruction to the first method (E.g. see Figure 3. PIC stub and rectangle display method and associated text) while the first slot is locked (as noted above in Claim 26, lock);”

“means for updating the polymorphic inline cache so that a second method of a second type will invoke a second polymorphic inline cache initialisation routine;” (E.g. see Figure 3. PIC stub and circle display method and associated text);

“means for making the first slot available to the first method;” (E.g. see Figure 3. PIC stub and associated text);

“means for calling the first method;” (E.g. see Figure 3. PIC stub and rectangle display method and associated text);

“means for locking the second slot of the polymorphic inline cache;” (as noted above in Claim 26, lock);

“means for filling the second slot with a call instruction to the second method (E.g. see Figure 3. PIC stub and circle display method and associated text) while the second slot is locked; (as noted above in Claim 26, lock)” and

“means for updating the polymorphic inline cache so that a Nth method of a Nth type will invoke a Nth polymorphic inline cache initialization routine.” (E.g. see Figure 3. PIC stub and associated text).

As Per claim 29, the rejection of claim 28 is incorporated and further Holzle discloses “the first, second, and Nth polymorphic inline cache initialization routines are identical.” (E.g. see Figure 3. PIC stub and associated text).

As Per claim 30, the rejection of claim 28 is incorporated and further the limitation “ the first polymorphic inline cache initialisation routine is called from a first thread simultaneously while an independently executing second or Nth thread is calling the second or Nth method or is calling the second or Nth polymorphic inline cache initialization routine.” is obvious in a multi-threading environment as noted above in Claim 26.

Art Unit: 2122

4. Claims 6-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Holzle in view of Bacon, further in view of Harriman US Patent No. 6,182,177.

As Per claim 6, the rejection of claim 1 is incorporated and further the combination of Holzle and Bacon do not explicitly discloses “said creating step (a) further comprises inserting a first illegal type value in a compare instruction of every slot of the polymorphic inline cache to indicate each slot is empty.”. However, Harriman, in analogous art, teaches Tracking of entries in the free token queue 115 may be accomplished by maintaining a list of available slots, or by maintaining an array of bits indicating the status of each slot in the command storage block 135. (E.g. see col. 3:42-46). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Harriman into the system of Holzle and Bacon, to indicate that each slot is empty. The modification would have been obvious because one of ordinary skill in the art would have been motivated to track of entries in the free token queue by maintaining a list of available slots (with array of bits indicating the status of each slot).

As Per claim 7, the rejection of claim 1 is incorporated and further the combination of Holzle and Bacon do not explicitly disclose putting a bit in each slot to indicate that each slot is empty. However, Harriman, in analogous art, teaches Tracking of entries in the free token queue 115 may be accomplished by maintaining a list of available slots, or by maintaining an array of bits indicating the status of each slot in the command storage block 135. (E.g. see col. 3:42-46). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Harriman into the system of Holzle and

Bacon, to indicate that each slot is empty. The modification would have been obvious because one of ordinary skill in the art would have been motivated to track of entries in the free token queue by maintaining a list of available slots (with array of bits indicating the status of each slot).

As Per claim 8, the rejection of claim 6 is incorporated and further the combination of Holzle and Bacon disclose “said locking step (e) further comprises replacing the first illegal type value in the compare instruction of the k slot with a second illegal type value to indicate the k slot is in use.” (E.g. see Bacon col. 3:37-51, i.e. thread identifier and “Bacon-bit”). However, Harriman, in analogous art, teaches Tracking of entries in the free token queue 115 may be accomplished by maintaining a list of available slots, or by maintaining an array of bits indicating the status of each slot in the command storage block 135. (E.g. see col. 3:42-46). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Harriman into the system of Holzle and Bacon, to indicate that slot is in use. The modification would have been obvious because one of ordinary skill in the art would have been motivated to track of entries in the free token queue by maintaining a list of available slots (with array of bits indicating the status of each slot).

As Per claim 9, the rejection of claim 1 is incorporated and further the combination of Holzle and Bacon do not explicitly disclose locking step (e) further comprises changing a bit in the k slot to indicate that the k slot is in use. However, Harriman, in analogous art, teaches Tracking of entries in the free token queue 115 may be accomplished by maintaining a list of available slots, or by maintaining an array of bits indicating the status of each slot in the

command storage block 135. (E.g. see col. 3:42-46). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Harriman into the system of Holzle and Bacon, to indicate that each slot is in use. The modification would have been obvious because one of ordinary skill in the art would have been motivated to track of entries in the free token queue by maintaining a list of available slots (with array of bits indicating the status of each slot).

As Per claim 10, the rejection of claim 8 is incorporated and further the combination of Holzle and Bacon disclose “said unlocking step (h) further comprises replacing the second illegal type value in the compare instruction of the k slot with a value of type k.” (E.g. see Bacon col. 3:37-51, i.e. thread identifier and “Bacon-bit”). However, Harriman, in analogous art, teaches Tracking of entries in the free token queue 115 may be accomplished by maintaining a list of available slots, or by maintaining an array of bits indicating the status of each slot in the command storage block 135. (E.g. see col. 3:42-46). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Harriman into the system of Holzle and Bacon, to indicate that slot is unlocked in the array of bits indicating the status of each slot. The modification would have been obvious because one of ordinary skill in the art would have been motivated to track of entries in the free token queue by maintaining a list of available slots (with array of bits indicating the status of each slot).

5. Claims 19-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Holzle in view of Bacon, further in view of Lee et al. "Reducing Virtual Call Overheads in a Java VM Just-in-Time Compiler", the 4th Annual Workshop on Interaction between Compilers and Computer Architectures, pp.21-33, 2000 (hereinafter Lee), further in view of Shimura US Patent No. 6,370,687.

As per Claim 19, the combination of Holzle and Bacon discloses "polymorphic inline cache implementing a lockable slot for each individual object type to a polymorphic call site in the application." as noted above in Claim 1. The combination of Holzle and Bacon do not explicitly disclose JIT compiler. However, Lee teaches evaluating the performance impact of Just-in-time (JIT) compilation and inline caches techniques in an actual Java virtual machine.. (E.g. see Abstract and associated text). Lee, in analogous art, teaches a computer system that has processor and memory with a JIT compiler that running a Polymorphic Inline Caches (E.g. see Page 23-26, Section 3.1.2 Polymorphic Inline Caches) programs. Lee, in analogous art, teaches "a central processing unit for executing an application;" (E.g. see Page 22, Section 2.2 1st paragraph, RISC machine, UltraSPARC); "memory connected to the central processing unit via a bus;" (E.g. see Page 22, Section 2.2 1st paragraph, RISC machine, UltraSPARC); "a just-in-time compiler for compiling object oriented applications for execution;" (E.g. see Page 22, Section 2.2 1st paragraph, LaTTe JIT Compiler). Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lee into the system of the combination of Holzle and Bacon, to CPU, memory and JIT compiler to execute PIC. The modification would have been obvious because one of ordinary skill in the art would

have been motivated to use a JIT compiler to test/verify PIC performance in a multithreading object oriented Java VM environment.

The combination of Holzle, Bacon and Lee does not explicitly disclose at least one input/output device connected to the bus and connected to a network interface to an external computer network. However Shimura discloses JIT compiler (E.g. see FIG. 2 and associated text) and network (E.g. see FIG. 2 and associated). Shimura, in analogous art, teaches “one input/output device connected to the bus and connected to a network interface to an external computer network”. (E.g. see FIG. 2 network interface 32 and associated text). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Shimura into the system of the combination of Holzle, Bacon and Lee, to have one input/output device connected to the bus and connected to a network interface to an external computer network. The modification would have been obvious because one of ordinary skill in the art would have been motivated so that in a compile server accepts the virtual machine computer programs from the network and compiles the same for the delivery to the requested clients.

As per Claim 20, the combination of Holzle and Bacon discloses “a polymorphic inline cache having a plurality of slots, each slot allocated to an object type of a method and locked to other object types of a polymorphic call site in the application” as noted above in Claim 1. The combination of Holzle and Bacon do not explicitly disclose JIT compiler. However, Lee teaches evaluating the performance impact of Just-in-time (JIT) compilation and inline caches techniques

in an actual Java virtual machine.. (E.g. see Abstract and associated text). Lee, in analogous art, teaches a computer system that has processor and memory with a JIT compiler that running a Polymorphic Inline Caches (E.g. see Page 23-26, Section 3.1.2 Polymorphic Inline Caches) programs. Lee, in analogous art, teaches “a processor for executing the application” (E.g. see Page 22, Section 2.2 1st paragraph, RISC machine, UltraSPARC); “memory connected to the processor with an internal bus” (E.g. see Page 22, Section 2.2 1st paragraph, RISC machine, UltraSPARC). Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lee into the system of the combination of Holzle and Bacon, to CPU, memory and JIT compiler to evaluate the PIC performance. The modification would have been obvious because one of ordinary skill in the art would have been motivated to use a JIT compiler to test/verify PIC performance in a multithreading object oriented Java VM environment.

The combination of Holzle, Bacon and Lee does not explicitly disclose a network interface to connect to an external computer network. However Shimura discloses JIT compiler (E.g. see FIG. 2 and associated text) and network (E.g. see FIG. 2 and associated). Shimura, in analogous art, teaches “a network interface to connect to an external computer network”. (E.g. see FIG. 2 network interface 32 and associated text). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Shimura into the system of the combination of Holzle, Bacon and Lee, to have a network interface to connect to an external computer network. The modification would have been obvious because one of ordinary skill in the art would have been motivated so that in a compile server

Art Unit: 2122

accepts the virtual machine computer programs from the network and compiles the same for the delivery to the requested clients.

Conclusion

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kuo-Liang J Tang whose telephone number is 703-305-4866. The examiner can normally be reached on M-F 8:30 to 5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q Dam can be reached on 703-305-4552.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306.

Kuo-Liang J. Tang

Software Engineer Patent Examiner


WEI Y. ZHEN
PRIMARY PATENT EXAMINER